International Journal of Electrical and Computer Engineering (IJECE)
Vol. 5, No. 3, June 2015, pp. 503~517
ISSN: 2088-8708

An Efficient Cache Organization for On-Chip Multiprocessor Networks

Medhat H Awadalla, Ahmed Sadek
Department of Electrical and Computer Engineering, SQU, Oman
Department of Communication, Electronics and Computers University of Helwan, Egypt
Department of Computer Science, Faculty of Computers and Information, Fayoum University, Egypt

Article Info

Article history:

Received Nov 18, 2014
Revised Apr 21, 2015
Accepted May 2, 2015

Keyword:

Chip Multi Processors
Interconnection mechanisms
Shared cache memory

ABSTRACT

To meet the growing computation-intensive applications and the needs of low-power, high-performance systems, the number of computing resources in single-chip has enormously increased. By adding many computing resources to build a system in System-on-Chip, its interconnection between each other becomes a challenging issue. This paper focuses on the interconnection design issues of area, power and performance of chip multi-processors with shared cache memory. It shows that having a shared cache memory contributes to the performance improvement; however, typical interconnection between cores and the shared cache using crossbar occupies most of the chip area, consumes a lot of power and does not scale efficiently with increased number of cores. This paper proposes an architectural paradigm in an attempt to gain smaller area occupation allowing more space for an additional cache memory. It also reduces power consumption compared to the existing crossbar architecture. Furthermore, the paper modified the typical MESI cache coherence algorithm to be tailored for the suggested architecture. The experimental results show that the developed architecture produces less broadcast operations compared to the typical algorithm.

Corresponding Author:

Medhat Awadalla
Departement of Electrical and Computer Engineering,
SQU, Oman
Email: medhatha@squ.edu.om

1. INTRODUCTION

The success of system-on-a-chip (SoC) hinges upon a well-concerted integrated approach from multiple disciplines, such as device, design, and application. From the device perspective, rapidly improving VLSI technology allows the integration of billions of transistors on a single chip, thus permitting a wide range of functions to be combined on one chip. From the application perspective, numerous killer applications have been identified, which can make full use of the aforementioned functionalities provided by a single chip. From the design perspective, however, with greater device integration, system designs become more complex and are increasingly challenging to design.

Multi-core processors have become the mainstream architectures as a result of their ability to provide parallelism and multi-tasking to achieve higher performance [1]. Initial multi-core designs replicated off-the-shelf cores multiple times, and then connected the cores together using an interconnection mechanism. These designs are called multi-core oblivious designs as they replicate cores unaware that each core is becoming part of a whole system. However in spite of the growing trend to put multiple cores on the die, a deep understanding is lacking in the literature of the design space of the interconnection framework, and particularly how it interacts with the rest of the multi-core architecture. For a given number of cores, the "best" interconnection architecture in a given chip multiprocessing environment depends on a myriad of

factors, including performance objectives, power/area budget, bandwidth requirements, technology, and even the system software. While interconnects are relatively well understood for connecting chips, multi-chip modules, and board-level nodes, connecting cores on the same chip presents a distinctly different problem. This is because power, area, latency, and bandwidth are all first-order design constraints for on-chip interconnects. Secondly, the design choices for the cores, caches, and interconnect interact to a much greater degree. For example, an aggressive interconnect design consumes power and area resources that then constrains the number, size, and design of the cores and caches. Thus, unlike a conventional multiprocessor, performance is not necessarily maximized by the highest bandwidth interconnect available. Of course, the converse is also true, that the number and type of cores (as well as on chip memory) also dictate requirements on the interconnect. In fact, increasing the number of cores places conflicting demands on the interconnect – requiring higher bandwidth while decreasing available real estate. Therefore, This paper addresses the interconnection design issues of area, power and performance of chip multi-processors with shared cache memory. On-chip networks architectures are believed to be the ideal solution for system on chip interconnection problems [2]. Networks on Chip (NoC) can improve design productivity by supporting modularity and reuse of complex cores, thus enabling a higher level of abstraction in architectural modeling of future systems [3]. Current NoC architectures consider the processing core and its cache memory as a single module.

Accessing cache memory is often a principal bottleneck in such multi-core systems, as multiple cores compete for on-chip limited memory resources, so cache memory organization and management is essential for improved performance and efficiency. Having shared cache that holds data accessible by multiple cores have proved to enhance the performance [4], but typical interconnection mechanisms have great impact on the system, especially typical crossbar that consumes a lot of area and correspondingly limit the available space for cache memory [5]. On-chip caching has been employed as a very effective approach to reduce memory bandwidth requirement. While caching helps to reduce off-chip memory bandwidth significantly, the effective use of caches is not always guaranteed. Useful content may be evicted due to address conflicts, which adds to the off-chip memory pressure. The problem is more significant in multi-core systems, in which multiple tasks are executing simultaneously. The contention of cache resources would potentially lead to significant inter-task cache interferences and thus much higher memory bandwidth requirements. Cache partitioning techniques have been a well-studied area in recent years. These cache organizations aim at improving cache utilization to achieve better performance and power efficiency [6, 7, 8]. Most existing cache partitioning techniques focus on reducing cache miss rates in order to increase system throughput or overall performance. However, the majority of these approaches do not consider off-chip memory bandwidth as a primary optimization goal.

This paper suggests an architectural model where cache memory is organized as both shared and private cache, and utilizing the new concept of NoC to implement the interconnection mechanism. The paper focuses on the 8-core processor as an example model where performance, area and power measures are studied and compared to the previously suggested models.

The rest of this paper is organized as follows: Section 2 gives the related work. Section 3 shows the typical interconnection mechanisms and their characteristics. Section 4 illustrates the cache organization of shared and private data. Section 5 demonstrates the network on chip (NoC) architecture. Section 6 presents the suggested model for 8-core processor. Section 7 concludes the paper.

2. RELATED WORK

As technology moves towards multi-core system-on-chips (SoCs), it becomes ubiquitous in all computing domains ranging from general purpose servers to the domain specific processors and from 3G cellular base stations to the latest game consoles [9]. These SoCs today have dozens of tiled cores on a single chip. The core count is expected to grow to hundreds or even thousands in the near future [10]. Conventional wisdom is to double the number of cores on a chip with each silicon generation [11]. For example, the latest release of NVIDIA Tesla C1060 GPU has as many as 240 cores integrated in a chip [12]. The fact that such a high number of cores will be tightly integrated onto the same die presents a fundamental challenge for on-chip communication among cores.

The network-on-chip (NoC) is an enabling technology for integration of large numbers of embedded cores on a single die. The existing method of implementing a NoC with planar metal interconnects is deficient due to high latency and significant power consumption arising out of long multi-hop links used in data exchange. Different networks-on-chip (NoCs) [13] are emerging as the scalable fabric for interconnecting the cores. They consist of routers, links, and well-defined network interfaces. One of the key issues in the design of NoCs is the reduction of both area and power dissipation. Such requirements impose important design choices like the topology, switching technique, routing algorithm and the architectural

implementation. As a result, most of current NoCs implement regular network topologies that can be easily laid out on a chip surface. Some topologies are preferred, since they offer lower power consumption than other topologies for application-specific mapping of tasks [14]. There have been several proposals and implementations of high-performance chip multi-processor architectures [15]. However, the latency, power consumption and interconnect routing problems of conventional NoCs can be addressed by replacing or augmenting multi-hop wired paths with high-bandwidth single-hop long-range wireless links. This opens up new opportunities for detailed investigations into the design of wireless NoCs (WiNoCs) with on-chip antennas, suitable transceivers and routers. Moreover, as it is an emerging technology, the on-chip wireless links also need to overcome significant challenges pertaining to reliable integration. In their paper, they present various challenges and emerging solutions regarding the design of an efficient and reliable WiNoC architecture. Interconnection mechanisms presented in [16] discussed the advantages and disadvantages of each mechanism. Cores in Hydra [17] are connected to the level 2 (L2) cache through a crossbar. Modularity results in enhanced control over electrical parameters and hence can result in higher performance or reduced power consumption. These interconnections can be highly effective in particular environments where most communication is local, explicit core-to-core communication. Due to their scalability, these architectures are attractive for a large number of cores. The crossover point where these architectures become superior to the more conventional interconnects studied is not clear, and is left for future work [18]. There is a large body of related work evaluating tradeoffs between bus-based and scalable shared memory multiprocessors, in the context of conventional (multiple-chip) multiprocessors. Some earlier implementations of the interconnection networks for multiprocessors have been described in [8,12]. However, on-chip interconnects have different sets of tradeoffs and design issues. Thus, the conclusions of those prior studies must be re-evaluated in the context of on-chip multiprocessors with on-chip interconnects.

Recent proposal of organizing cache memory based on data sharing are presented in [19]. Architecture in Nahalal model [20] partitions the L2 cache space according to the programs' data sharing, and can thus offer vicinity of reference to both shared and private data. The organization of cache memory in [20] is based on non-uniform cache architecture (NUCA) array with a switched network embedded in it for high performance.

3. TYPICAL INTERCONNECTION MECHANISMS

This section discusses interconnection mechanisms which are used to connect between multi-core processor elements

a. Nahalal: Cache Organization

In monolithic processor architectures, cache memory is organized according to contents (e.g., instructions cache and data cache) or hierarchically (e.g., level 1 and level 2 cache). In multi-core environment, other factors can efficiently organize the cache memory in order to gain the benefits of having multiple cores on the same chip. For instance, data sharing is an important factor to consider for multi-core systems, where there is a discrimination between shared data that is accessible by multiple cores and private data that is accessed by a single core. As global wire delays become a dominant factor in VLSI design [13], on chip access time depends on the distance between the processor and the data. In many designs, L2 cache is typically located as a bulk in one location (e.g., at the center surrounded by the processors) and hence producing a non-uniform access across the chip. In commercial workloads, significant fraction of memory accesses involve shared data that cannot be replicated without performance penalty to maintain coherence.

The newly proposed Nahalal cache organization in this paper is inspired from the layout of a cooperative village called Nahalal, shown in Figure 1.a which is based on urban design ideas from the 19th century [20]. In Nahalal, public buildings are located in an inner circle, enclosed by a circle of homesteads. Private areas of land are arranged in outer circles, each close to its owner's house.

Figure 1.a. Nahalal village layout

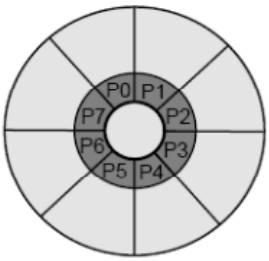

Figure 1.b. Conceptual layout

The suggested architecture is similar to the organization of the village where a fraction of L2 cache memory is located at the center of the chip, shared and surrounded by the processing cores. The rest of L2 cache is placed on the outer circle surrounding the cores as shown in Figure 1.b. Implementation of Nahalal improves L2 cache access times by up to 41% compared to traditional CMP designs.

3.2. Networks on Chip (NoC)

NoC architectures may adopt design concepts from traditional computer networks, however on chip implementations of networks need different considerations, where network architectures and protocols have to deal with the advantages and limitations of the silicon fabric. On chip implementations have advantages over traditional computer networks like local proximity and deterministic nature where nodes to be connected are determined before the network is designed. Moreover, dynamic changes of links (link upgrades or failures) are not expected on-chip. Also, highly reliable link operation can be assumed.

Based on these considerations, the modules are interconnected by a network of multi-port switches connected to each other by links composed of parallel point-to-point lines. The network, for example, applies a mesh topology and employs wormhole packet forwarding with hop-by-hop credit-based backpressure flow control (for lossless buffer operation and minimal buffer requirements) [6]. Packets are forwarded based on a shortest path, employing X-Y mechanism where a packet is routed first in the "X" direction, then in a perpendicular direction. This scheme leads to a simple, cost-effective router implementation. The packet is divided into multiple flits [15]. The flits are classified as the following types:

- FP (Full Packet): A one-flit packet
- EP (End of Packet): Last flit in a packet
- BDY (Body): A non-last flit

So, all flits except the last one in a packet are tagged BDY. The first flit of a packet can be detected as the first flit following either a FP or EP flit and this identification triggers the routing mechanism. Every arriving flit of a packet on the input port of the router is stored in an input buffer. On the arrival of the first flit, the routing algorithm determines the output port that the packet destined. Routing information per input port is stored in the current routing table (CRT), until the tail flit of the packet is received, processed and delivered. When a flit is forwarded from an input to an output port, one buffer becomes available and a buffer-credit is sent back to the previous router. Each output port of a router is connected to an input port of a neighbor router. The output port maintains the available flit slots in the buffer of neighbor input port. This number is stored in the Next Buffer State (NBS) that is decremented upon transmission of a flit and incremented upon the reception of a buffer credit from the neighbor router [16].

3.3 NOC Connecting Shared Cache Memory

In this section, we focus on an 8-core processor with 8 cache banks. Crossbar as an interconnection mechanism carries a heavy area overhead. If the total die area is around 400 mm2, then the area overhead for an acceptable latency is around 46.8% for full sharing (nearly half the chip!). For calculating on-chip memory sizes, it is determined to be 1 bit per square micron, or 0.125MB per mm2 for 65 nm technology [2]. If the cache banks were private, and an SBF is used to interconnect these banks to the 8 cores, there would be 3MB of cache memory available for each core. The proposed architectural model in Figure 2 is inspired from the Nahalal cache architecture where part of the L2 cache memory is shared and placed in an inner circle surrounded by the NoC routers. Each router is connected to neighbor routers, a processing core, and a cache memory bank. The remaining part of the L2 cache is kept private for each core.

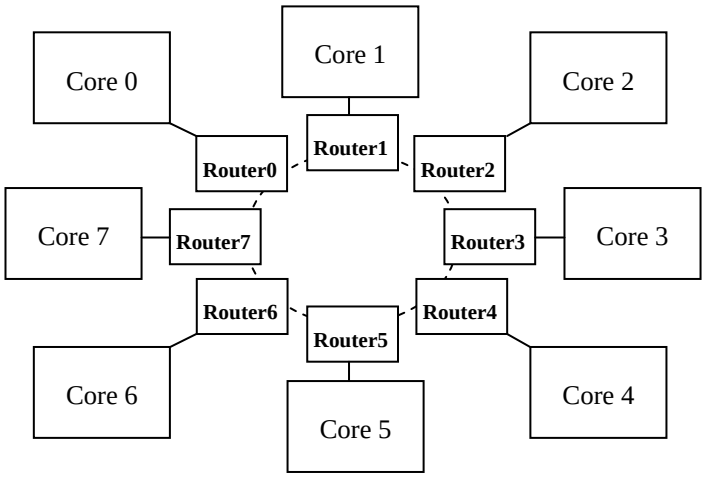

Figure 2. NoC connecting shared cache memory

By calculating the area occupied by the on chip network, we can calculate the remaining area available for cache memory. The area occupied by links can be calculated according to:

$$Cost_{wire\text{-}area}=A_0 \sum_{i\ links} W(i)L(i) \quad (1)$$

Where A0 is the wire pitch, W(i) is the i link width (number of wires) and L(i) is the length of link i [2]. Considering 16 bit data signals from one port to another, then, the total area occupied by 24 links that are implemented in 8X metal plane with a wire pitch of 1.6 µm (for 65 nm technology) is estimated to be ~3 mm2 [2].

The area cost of router is affected by several parameters, number of ports and size of flit buffers. A good estimation for the area cost of the router is flip-flop count [16]. The cost of a router estimated by flip-flop count results in ~ 625 flip-flops taking an area of about 0.103 mm2 (for 65 nm technology). So, we need about 0.825 mm2 for the 8 routers, and hence, 3.825 mm2 for the entire network. This area occupied by the network would limit the available cache space to be 2.94 MB per core. If we allocate 2MB to each core as a private cache, we have a remaining 7.52 MB of cache memory to be shared by the 8 cores at the inner circle of the architecture. Therefore, the area required to build the NoC is far smaller than that needed to construct the crossbar. Figure 3 shows the area taken by the crossbar interconnection for 2-way, 4-way, and 8-way sharing for the 8-cores processor [2].

By increasing the number of data lines, we can achieve a higher bandwidth, but with increased area overhead. By utilizing Eq. (1), the network area overhead for different number of data lines is demonstrated in Figure 4. Figure 5 shows the correspondingly available cache memory space versus the number of data lines. From these figures, it is observed that the area occupied by the network even with 256 bits of data is much smaller than that of the crossbar (nearly 15%). Dynamic power dissipation in switching circuits is [16]:

$$P_d=C_L V_{dd}^2 f \quad (2)$$

where CL is the load capacitance, Vdd is the supply voltage, and f is the switching frequency. Load capacitance consists of wire capacitance and gate capacitance of the driven transistor, assuming that the wire capacitance is the dominant and estimated to be 0.2 pF/mm [13], and link frequency of 2.5GHz, then the dissipated power per link would be about 46mW giving a total of 1.1W for all links. The dynamic power per flip-flop is estimated to be 0.05mW at 2.5GHz for 65nm technology, and then the total power dissipated by the routers would by about 250mW. The power overhead due to a typical crossbar is very significant. The overhead can be more than the power taken up by three full cores for a completely shared cache (more than 25W). Figure 6 shows the power dissipated by the crossbar for various sharing level for the 8-cores processor [2].

Increasing the number of data lines to get a higher bandwidth affects in turn the power dissipated by the network. Figure 7 shows the power dissipated in the network for different number of data lines. From

these results, it is obviously that even with 256 data lines, the power dissipated by the network is about 60% of that dissipated by the crossbar.

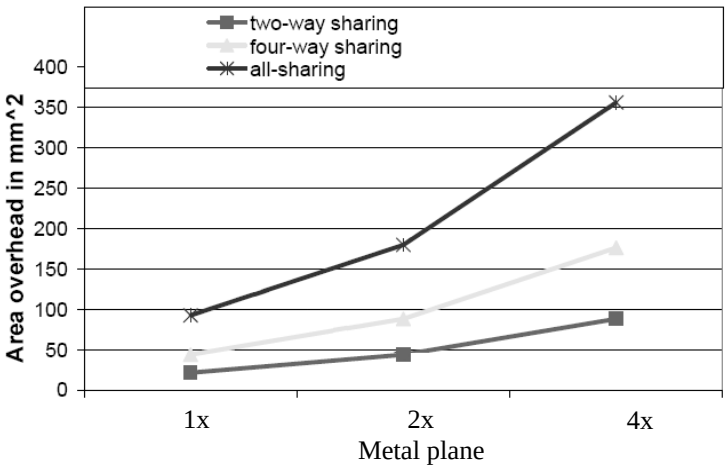

Figure 3. Area overhead of crossbar

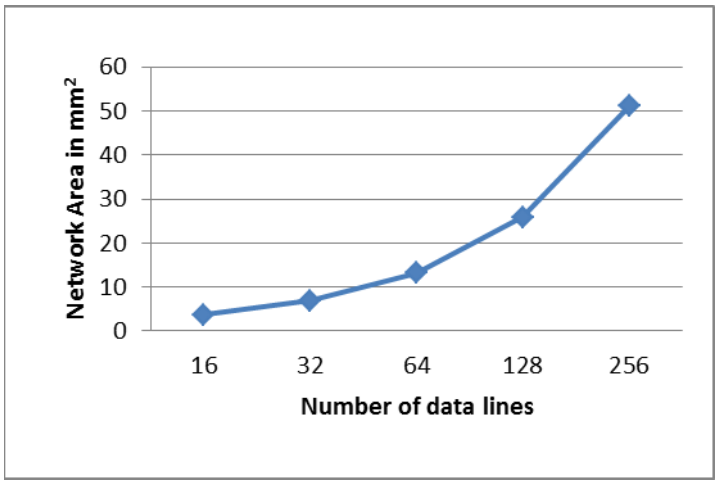

Figure 4. Relationship between bandwidth and area

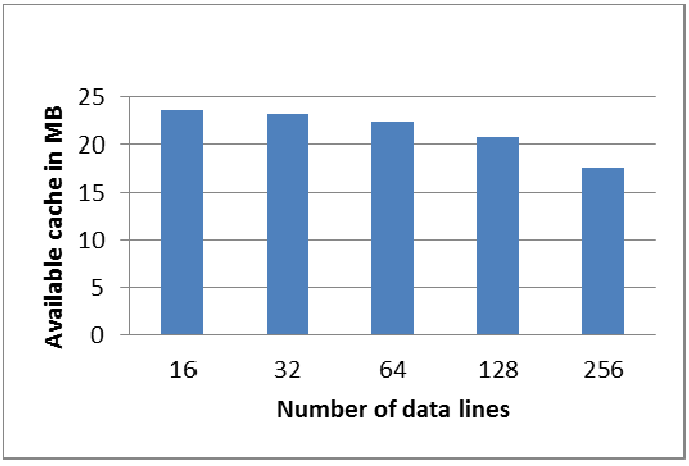

Figure 5. Available cache space for different number of data lines

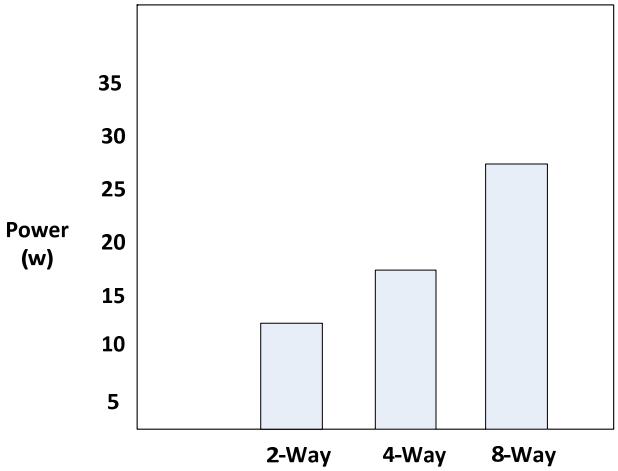

Figure 6. Power dissipated by the crossbar with various level of sharing

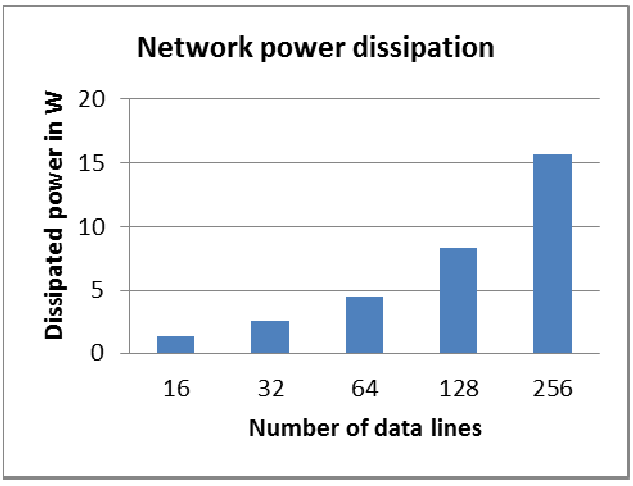

Figure 7. Power dissipated for various numbers of data lines

3.4 Effect of Adding a Customized BUS to the NOC

Interconnect architectures which rely solely on NoCs have several drawbacks like latency of critical signals and complexity of operations that require global knowledge or control. For example, the distributed nature of the network is an obstacle for a broadcast operation.

A broadcast operation on a network requires additional massive duplication of unicast messages. A recent research suggested the addition of a low latency, customized shared bus to the NoC [17]. This bus is inferior to the NoC in terms of bandwidth, but it has an inherent main property: it is capable of broadcasting information. This newly suggested architecture is called Bus Enhanced Network on Chip or "BENoC".

The addition of a shared bus will subsequently contribute to the added interconnect overhead in terms of area and power. But even with the added overhead of the shared bus, the total overhead of the BENoC is still lower than that of the crossbar interconnect.

The shared bus consists of 7 bytes width address bus, 12 bytes width snoop bus and 8 bytes width response bus. These widths are typical as suggested for the 8 core case in [2]. If the shared bus is to use the low latency 8x metal layer, we can utilize equation (1) to calculate the area consumed by the bus. Figure 8 shows the area consumed by the BENoC and confirms that the total area of interconnect, even with a 256 bits of data lines, is still not comparable with the area overhead of the crossbar implementing all way sharing in the 4x plane.

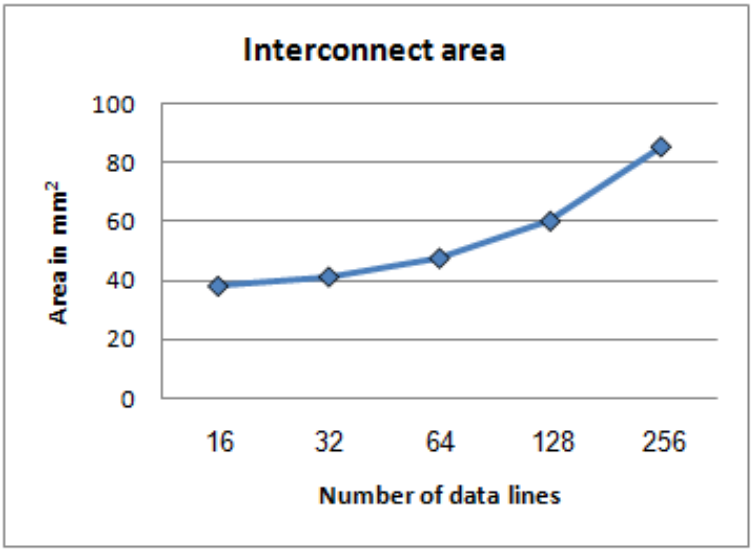

Figure 8. Relationship between bandwidth and area of BENoC

Taking into account the added power consumption of the shared bus that spans all the width of the chip, we can reuse equation (2) to calculate the power consumed by the BENoC interconnect. Figure 9 shows that the power dissipated by the interconnect even with 256 bits of data lines is still less than that consumed by the crossbar implemented in 4x plane (about 80%).

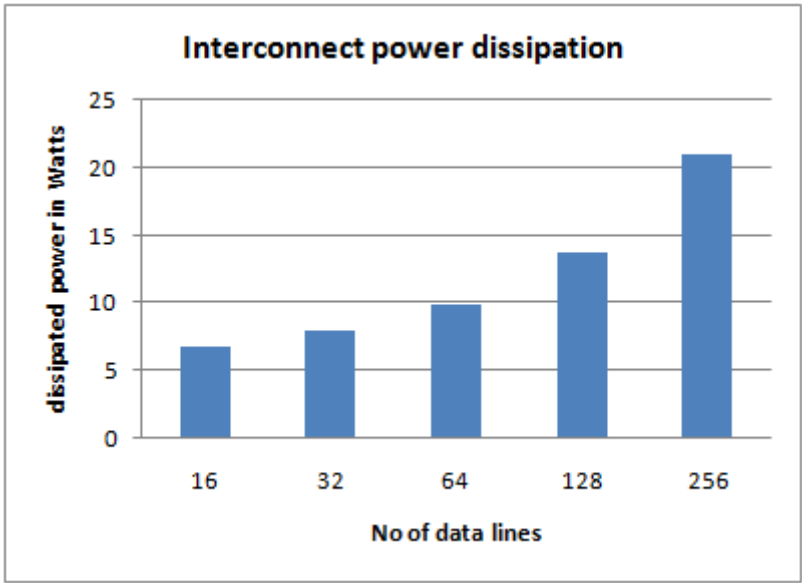

Figure 9. Power dissipated for various numbers of data lines for BENoC

4. CACHE COHERENCE WITH MESI WRITE BACK PROTOCOL

In a shared memory multiprocessor system with a separate cache memory for each processor, it is possible to have many copies of any one instruction operand: one copy in the main memory and one in each cache memory. When one copy of an operand is changed, the other copies of the operand must be changed also. Cache coherence is the discipline that ensures that changes in the values of shared operands are propagated throughout the system in a timely fashion. In the following subsections the typical MESI cache coherence algorithm will be discussed.

4.1. Typical MESI Coherence Algorithm

Cache coherence is a must to avoid using inconsistent data and therefore, there must be some means to distribute information about changes between cache memory and main memory.

The idea in write back cache coherence protocols is to hold a line in the cache, even if it is modified, as long as other nodes do not need it, and update contents externally only when another node needs it. The process in which one node notifies the other nodes of its actions and other nodes listen is called snooping [19]. The MESI model specifies that each cache line has two status bits, so, each line can be in one of the following possible states:

Modified: the cache line resides only in this cache and its contents are modified relative to the main memory.

Exclusive: the cache line resides only in this cache and its contents are the same as memory.

Shared: the cache line is shared with others.

Invalid: the cache line contains no valid memory copy.

Modified and exclusive lines are owned by the cache and can be changed without telling others. However, when attempting to access non-owned line, the action is to be broadcasted and if the line is owned by another cache, the owner is to update the world and change its line state. If we apply a sample access sequence on 8 core CMP, we can see the importance of broadcasting to maintain coherence between nodes and figure out the performance of this kind of operations. Assume beginning from reset state; let's evaluate the following sample sequence:

Core 0 reads a0: broadcast address to all nodes and the line status is now "E".

Core 0 reads a0: core 0 reads a0 from cache and the status still "E".

Core 0 writes a0: core 0 modifies a0 only in its cache only and status is now "M".

Core 1 reads a0: core 1 broadcasts the address, core 0 broadcasts a response and supplies data to both cache and main memory and status changes to "S".

Core 0 reads a0: core 0 reads from cache, status still "S".

Core 1 writes a0: core 1 broadcast address so that other nodes invalidate their copies and status changes to "E".

Core 1 writes a0: core 1 modifies a0 in its cache and changes status to "M".

Core 0 reads a0: core 0 broadcasts the address, core 1 broadcasts a response and supplies data to both cache and main memory and status changes to "S".

The previous sequence needed to broadcast information 6 times to all 8 processing cores consuming 24 steps if performed using the on chip network, besides that, the status of the cache line a0 has been changed to "S" twice, first at step 4, and second at step 8, and there are two copies of a0 in both cache memories of the two processing cores.

4.2. The Proposed Tuned-MESI Coherence Algorithm

The main concept of Nahalal cache organization is to place the shared information in a shared location without duplication at every core, therefore, when a cache line is shared between one or more processing cores, it is migrated to the shared cache memory so that it is accessible by all cores. This cache line is considered a hot line and can be accessed by all cores.

To realize this concept, the traditional MESI cache coherence algorithm must be modified to suit the new architecture taking into account the existence of shared and hot shared cache lines. This modified algorithm is called Tuned-MESI. The traditional MESI algorithm is still applicable for the new architecture before a line is considered hot. The flowchart and state diagram of tuned-MESI algorithm is depicted in Figures 10 and 11.

Going through the same sequence again for the suggested architecture that has an advantage of a shared bus that all nodes snoop so that broadcasting is performed much more efficiently, and implements the Tuned-MESI algorithm. Assume that a shared line is considered hot after two external accesses.

Core 0 reads a0: broadcast address to all nodes and the line status is now "E".

Core 0 reads a0: core 0 reads a0 from cache and the status still "E".

Core 0 writes a0: core 0 modifies a0 only in its cache and status is now "M".

Core 1 reads a0: core 1 broadcasts the address, core 0 broadcasts a response and supplies data to both cache and main memory and status changed to "S".

Core 0 reads a0: core 0 reads from cache, status still "S".

Core 1 writes a0: line is migrated to shared area and considered hot line. Information about the Hot Line is broadcasted to all cores and status is "S".

Core 1 writes a0: core 1 modifies a0 in shared cache and status still "S".

Core 0 reads a0: core 0 reads from shared cache and status still "S".

Analysis of the previous sequence shows that broadcasting operations have been reduced to 4 operations. Beside the facts that these broadcasts have been carried out using the shared bus that requires less cycles to propagate a message to all processing cores compared to the NoC interconnect alone.

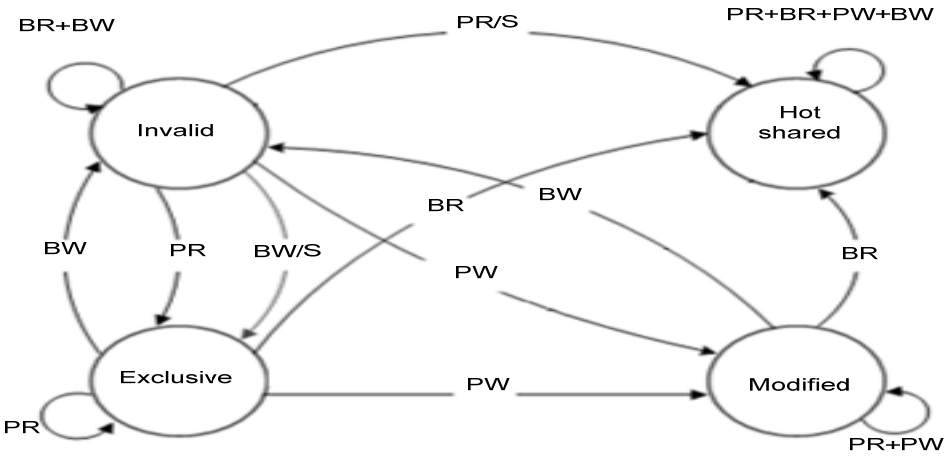

Figure 10. State transition diagram of Tuned-MESI

5. SIMULATED EXPERIMENTS AND DISCUSSION

The program models two processors each of them is consisting of 8 cores. The first processor has only private caches and implements the typical MESI algorithm to maintain cache coherence between all 8 cores. The second processor uses the proposed architecture that organizes cache memory as both shared (at the inner circle) and private (private cache for each core). This processor implements the proposed Tuned-MESI algorithm to maintain cache coherence. The program performs a random sequence of cache line accesses and measures the number of broadcast operations needed to maintain cache coherence for the processor that implements the typical MESI algorithm, and for the proposed architecture that implements the Tuned-MESI cache coherence algorithm. The user defines the following parameters:

Number of participating cores: cores that will be participating in the access sequence.
Number of modeled cache lines: cache lines that will be involved in the access sequence.
Threshold : number of accesses after which a line is considered Hot.
Hot space: available space for Hot Shared cache lines.
Private lines: percentage of total number of lines that will be private for cores.

The "Threshold" and "Hot space" parameters are only applicable for the proposed architecture. They have no effect on the typical algorithm.

The simulator is set to generate 100 cache lines. To measure the broadcasts incurred by the access of a random core to a random cache line with a random read or write access, we used a sequence length of 100,000 accesses. The user variable parameters are manipulated to demonstrate different factors affecting the performance of the architecture. For a configured set of parameters, the simulator is run 10 times and the results are averaged. The effect of following parameters (participating cores, threshold, hot space and the level of sharing) on the behavior of the processors is addressed. For a small threshold of two, hot space of five and level of sharing (10% of all lines are shared).

From the achieved results shown in Figure 12, it is obvious that when the number of participating cores increases, the number of broadcasts increases. The tuned-MESI outperformed the typical algorithm because the available shared cache was relatively enough to accommodate the small percentage of shared lines. If the percentage of shared lines is increased to 50%, the typical algorithm shows the same profile where the number of broadcasts increases as the number of shared lines and participating cores increases as illustrated in Figure 13. However, the number of broadcasts was significantly increased as the number of shared lines has increased, which means more broadcasting.

The performance of the Tuned-MESI algorithm was not reasonable, because the available hot space is relatively small to accommodate the shared lines. An increased broadcast overhead was incurred to remove the least accessed hot lines from shared cache and add others to it. By increasing the threshold of the cache line to be considered hot to ten, again the performance of the tuned algorithm has been improved by

increasing the threshold value as clear in Figure 14. The typical algorithm also shows the same performance with the number of broadcasts increases as the number of shared lines and participating cores increases.

Furthermore, the concept of migrating hot lines into the congested shared area by removing the least accessed cache line was a consistent technique. The hot cache area holds the most accessed hot shared cache lines. By increasing the available cache space for hot cache lines to 40, so it can accommodate 80% of the shared cache lines, and as mentioned before, the hot space size and the threshold value do not affect the typical algorithm, and the typical MESI shows the same increasing profile as the number of participating cores and level of sharing is increased as shown in Figure 15.

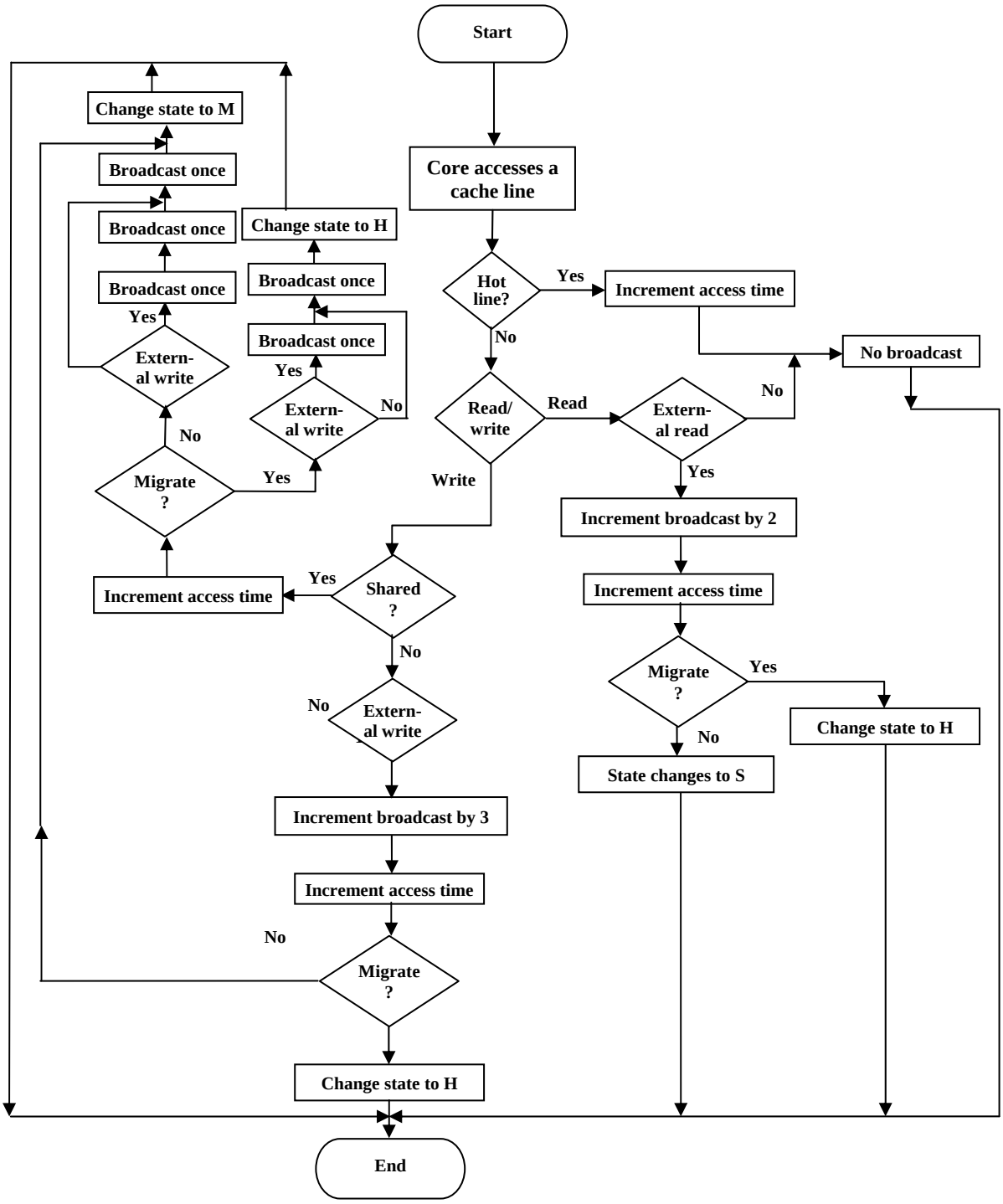

Figure 11. Flow chart of Tuned-MESI

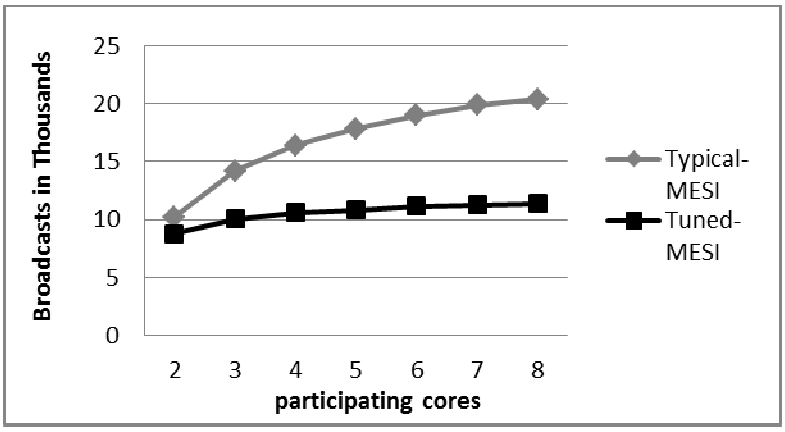

Figure 12. Performance evaluation for low values of threshold, hot space, and level of sharing

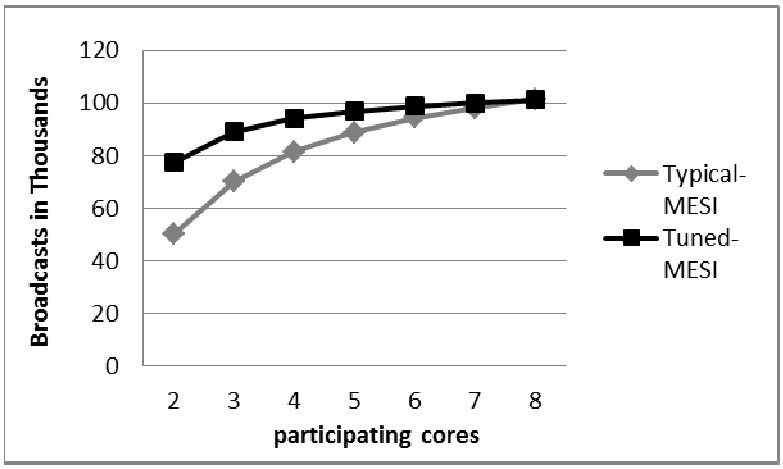

Figure 13. Performance evaluation for increased percentage of shared lines

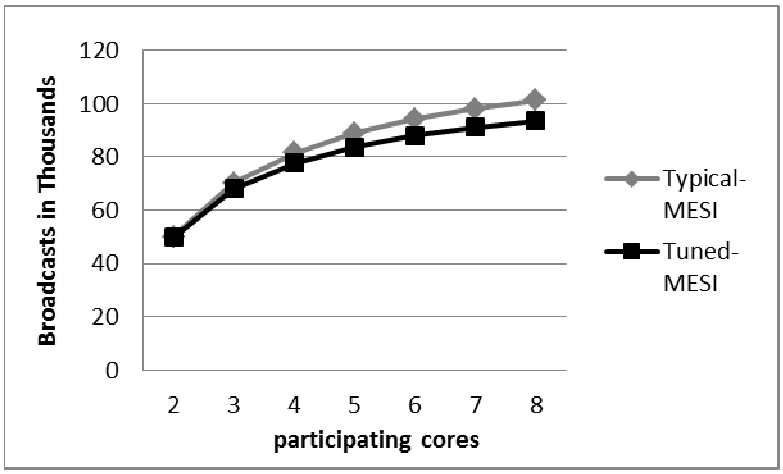

Figure 14. Performance evaluation with increased threshold

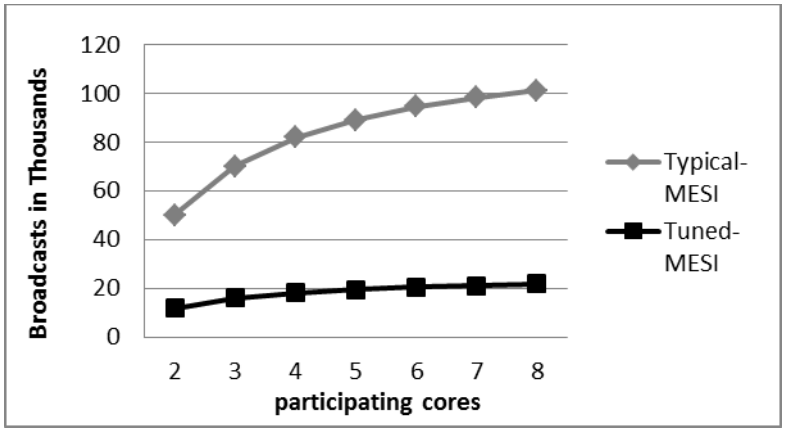

Figure 15. Performance evaluation with increased hot space

However Tuned-MESI algorithm performs much better than the typical one. As the level of sharing is increased, and the shared cache space can accommodate 80% of the shared cache lines, the amount of broadcasting is reduced. By increasing the available space so that it can accommodate all the shared cache lines, and decreasing the threshold such that hot lines are migrated as soon as it is accessed by two cores. As depicted in Figure16, the typical algorithm cannot be exhibited because of the great difference in range between the two algorithms under the specified conditions. Typical algorithm shows the same increasing profile as previously occurred, it is not affected by the hot space or the threshold value as shown in Figure 17.

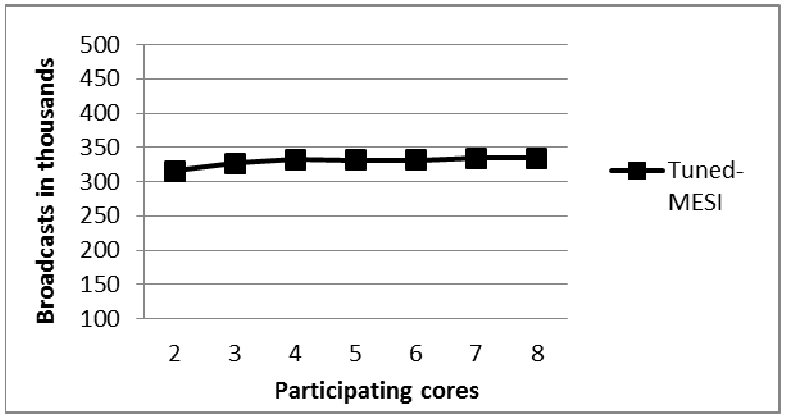

Figure 16. Performance evaluation with increased hot space and decreased threshold

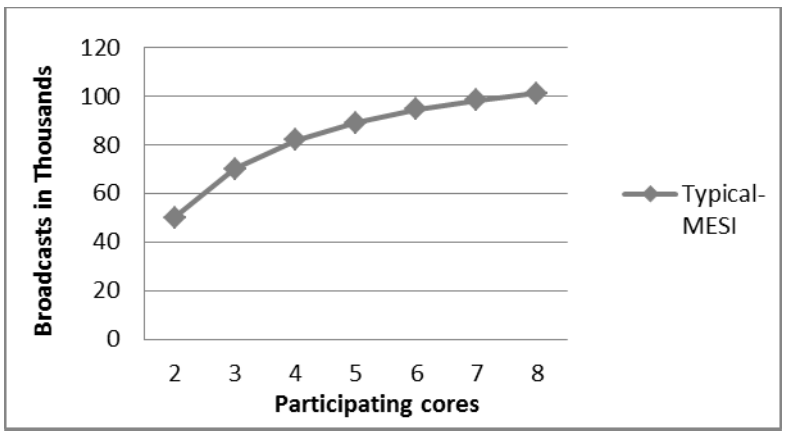

Figure 17. Performance evaluation of typical MESI

While Tuned-MESI algorithm dramatically outperforms the typical one. It also shows that the tuned algorithm is slightly affected by the increased number of participating cores. However, these results were achieved because of the optimistic assumption that the shared cache memory can accommodate all hot shared cache lines.

Based on all achieved remarkable results, the Tuned MESI algorithm is more reliable and efficient. Even with a small available cache space, it can perform considerably well by increasing the threshold value and migrating only the hot shared lines. This performance improvement was accomplished due to the proposed algorithm that takes advantage of the special organization of the proposed architecture offering a space for shared data, and a space for private data.

6. CONCLUSION

Multi-core processors are the next step in the processors performance growth. However, fundamental questions remain regarding methodology, and implementation for multi-core designs. Typical interconnection mechanisms are resources hungry in terms of area occupation and power dissipation, especially when having shared cache memory between cores. Networks on Chip are the future interconnection scheme for future designs that will scale as the number of cores on the chip increases and yet, offer a less area and power overhead. Cache memory organization and management is an essential aspect to consider when designing a multi-core processor. The concept of having both private memory for private data and shared memory for shared data is proved to be adequate.

This paper suggests a new architecture that organizes the cache memory as both private and shared memory and utilizes Network on Chip as the interconnection mechanism. The proposed interconnection architecture achieves smaller area occupation allowing more space to add additional cache memory and also achieves better power consumption compared to the existing architectures. Cache memory coherence is a factor that should be taken into account for multi-core systems and the typical cache coherence algorithm will not be suitable for the proposed architecture. A new cache coherence algorithm called Tuned-MESI is suggested to maintain consistency between cache memory units for the suggested architecture. This algorithm remarkably minimized the inter-core communication cost compared to the typical MESI algorithm.

REFERENCES

[1] S. Deb, A. Ganguly, P. Pande, B. Belzer, and D. Heo, "Wireless NoC as Interconnection Backbone for Multicore Chips: Promises and Challenges", *IEEE journal on emerging and selected topics in circuits and systems*, vol. 2, no. 2, pp. 228-239, June 2012.

[2] A. Golander, N. Levison, O. Heymann, A. Briskman, M.J. Wolski, and E.F. Robinson, "A Cost-Efficient L1–L2 Multicore Interconnect: Performance, Power, and Area Considerations", *IEEE transactions on circuits and systems*, vol. 58, no. 3, pp. 529-538, 2011.

[3] P. Abad, V. Puente, P. Prieto, and J. A. Gregorio, "Rotary Router: An Efficient Architecture for CMP Interconnect Networks", *In Proceedings of the 34th Annual International Symposium on Computer Architecture*, pp. 116-125, 2007.

[4] J. Zhan, N. Stoimenov, Y. Ouyang, L. Thiele, V. Narayanan, and Y. Xie, "Designing energy-efficient NoC for real-time embedded systems through slack optimization", *In Proceedings of ACM/IEEE Design Automation Conference (DAC)*, 2013.

[5] H. Shekofteh, and M.B. Khalkhali, "Reducing cache contention in a multi-core processor via a scheduler. Advanced Computer Theory and Engineering", *3rd International Conference on Advanced Computer Theory and Engineering (ICACTE)*, pp. 555-558, Aug. 2010.

[6] J. Huh, C. Kim, H. Shafi, L. Zhang, D. Burger, and S.W. Keckler, "A NUCA Substrate for Flexible CMP Cache Sharing", *IEEE Transactions on Parallel and Distributed Systems*, vol. 18, no. 8, pp. 1028-1040, 2007.

[7] S. Cho and L. Jin, " Managing Distributed, Shared L2 Caches through OS-Level Page Allocation", *In Proceedings of the 39th Annual IEEE/ACM International Symposium on Microarchitecture*, pp. 1-11, Dec. 2006.

[8] A. Vidapalapati et al., "NoC architectures with adaptive code division multiple access based wireless links", in Proc. IEEE Int. Symp. Circuits Syst. (ISCAS), May 2012.

[9] Chu E.T. and W. Lu, "Cache utilization-aware scheduling for multicore processors", *IEEE Asia Pacific Conference on Circuits and Systems*, pp.368-371, Dec. 2012.

[10] T. Farhat, G. Yvette, A. Caaliph and P. Jean-Marc, "An efficient and flexible hardware support for accelerating synchronization operations on the STHORM many-core architecture", *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 531 – 534, 2013.

[11] K. Rakesh, " Holistic Design for Multi-core Architectures", PhD thesis, University of California, San Diego, 2006.

[12] J.L. Shin et al., "The next generation 64b SPARC core in a T4 SoC processor", *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech Papers*, pp. 60–62, 2012.

[13] T. Ye, "On Chip Multiprocessor Communication Network Design and Analysis", PhD thesis, Stanford university, 2003.

[14] Y. Ouyang and Y. Xie, "LOFT: A High Performance Network-on-Chip Providing Quality-of-Service Support", *Proceedings of Intl. Symp. on Microarchitecture (MICRO 2010)*, pp. 351-356, 2010.
[15] T. Tao Ye and G. Micheli, "Physical Planning for On-Chip Multiprocessor Networks and Switch Fabrics", In *Proceeding of the14th IEEE International Conference on Application-Specific Systems, Architectures and Processors (ASAP'03)*, pp. 97-107, 2003.
[16] I. Cidon and I. Keidar. "Zooming in on Network on Chip Architectures", *Technical Report CCIT 565, Technion Department of Electrical Engineering, Israel Institute of Technology*, 2005.
[17] J. Chang and G.S. Sohi, "Cooperative Caching for Chip Multiprocessors", *In Proceedings of the 33nd Annual International Symposium on Computer Architecture*, pp. 264-276, June 2006.
[18] R. Ho, K. Mai, and M. Horowitz, " The future of wires", *Proceedings of IEEE*, vol. 89, issue 4, pp. 490-504, 2001.
[19] Z. Chishti, M.D. Powell, and T.N. Vijaykumar, "Optimizing Replication, Communication, and Capacity Allocation in CMPs", *In Proceedings of the 32nd Annual International Symposium on Computer Architecture*, pp. 1-12, June 2005.
[20] G. Zvika, K. Idit, K. Avinoam, and W. Uri, "Nahalal: Cache Organization for Chip Multiprocessors", *IEEE Computer Architecture Letters*, vol. 6, no. 1, pp. 21-24, 2007.